Cross Comparison of Multiple Machine Learning Implementations for Poem Dating

Jiachen Liu, Simon Harris, Will Tokunaga

Abstract—This report summarizes a cross comparison of multiple machine learning implementations for predicting the century in which a piece of poetry was written. The implementations analyzed poetry on 1) sentence structural level using traditional feedforward neural networks 2) word choices level using traditional feedforward neural networks, naïve bayes, linear support vector machine 3) word sequences level using vanilla recurrent neural networks, long short term memory recurrent neural networks, and bidirectional long short term memory neural networks. Highest performance was achieved by naïve bayes, linear SVM, and neural networks that analyzed word frequencies and choices. The results indicated that word choices were most indicative of the century in which a poem was written and all recurrent neural networks achieved suboptimal performance.

Introduction

With the rapid advancement in the field of machine learning, an increased variety of machine learning algorithms have been developed and applied in a wider range of fields. One area of machine learning that have gained increased attention and popularity is natural language processing (NLP), which aims to program machines to interpret human language and interact with human using natural languages. Some common examples of NLP include machine translation, speech recognition, chatbots and sentiment analysis.

There are multifarious machine learning algorithms that can be used for NLP tasks, with common ones being artificial neural networks (ANN), naïve bayes and support vector machine (SVM). The choice of the algorithm depends on the specific dataset and although ANN seem to be quite robust in general based on our prior lab explorations, the optimal algorithm can be hardly concluded without a cross comparison of performance on the specific task.

Based on our interest in poetry and NLP, we conducted a group project that examined different machine learning implementations to date poems. Our implementation analyzed poems written in 17th, 18th, 19th centuries from three different angles: sentence structures, word frequencies/choices, and word sequences. The algorithms we implemented included naïve bayes, linear SVM, and different types of ANNs. Specifically, we used attributes including word diversity, poem length, average word length, word sentiment, and part of speech proportions along with traditional feedforward neural networks to examine the predictiveness of sentence structures in dating poems. We used 1) traditional ANN with relative frequencies of high occurrence words as attributes, 2) ANN with word embeddings 3) naïve bayes 4) linear SVM with TF-IDF word vectorization to examine the predictiveness of word frequencies/choices in dating poems. Finally, we examined the predictiveness of word sequences in poem dating using sequence prediction approaches implemented by recurrent neural networks (RNN), long short term memory recurrent neural networks (LSTM), and bidirectional long short term memory neural networks (Bidirectional LSTM) with word

embeddings.

To our surprise, ANN with word frequency attributes, naïve bayes and linear SVM achieved highest accuracy (~ 80%), followed by bidirectional LSTM (~ 70%), ANN with word embeddings (65%-70%), LSTM (55% ~ 65%), and RNN (~37%). This suggest a higher predictiveness of word choices over word sequences in dating poems for our particular dataset and a demonstration of Occam's razor's concept that simpler models sometimes can achieve better performance than those more complicated ones.

Background and related works

Natural language processing (NLP) is a large field in machine learning, and many different kinds of techniques have been found to be effective for analyzing text. One review paper goes over the concepts behind different techniques used in NLP, and where NLP is going in the future[5]. As far a poetry goes, one group of researchers decided to use a variety of machine learning techniques to analyze the meter of English poetry[1]. Some of those techniques were ones that we also used, like naive Bayes and support vector machines, and other techniques included the averaged perceptron and hidden Markov models. Similar to some of the neural network models that we used, they used attributes like word length and part of speech tags to feed into their models. Another team looked at the emotional content of Arabic poems, also using naive Bayes and SVMs along with other techniques such as hyperpipes[2]. Most similarly to our project, one blog writer used a naive Bayes classifier to predict the centuries poems were written in.[3] Despite the similarity to our problem, we thought it was still worthwhile to compare the techniques we used to the one they used, and figure out which one is best.

Problem

In our experiment, we used multiple machine learning algorithms to predict when a piece of poetry was written. The poetry dataset was obtained from an open source GitHub project that used naïve bayes to date poems. After removing the largely underrepresented 16th century poem entries, the remaining dataset consist of poems written in 17th, 18th, and 19th centuries, with a total of 4067 pieces of poems. The summary statistics of proportion of each label is given in Table1.

Table 1. proportion of poems from 17th,18th,19th centuries

	# of Instances (percentage)
17th	1426 (35%)
18th	1435 (35%)
19th	1197 (30%)
Total:	4057

Solution

We implemented multiple machine learning algorithms to extract poem attributes from three different angles (sentence structures, word frequencies, and word sequences) to predict which century a given piece of poem was written.

Sentence structure level:

We used diversity of word choices (number of unique words normalized by poem length), poem length, average word length, overall poem sentiment (measured by TextBlob library), and part of speech proportions (proportion of nouns, verbs, adverbs, etc.) as attributes and used traditional feedforward neural network as the learning algorithm.

Word frequency level:

We implemented this with multiple learning algorithms

1) We used the relative frequency of top k most common words as attributes and used a simple ANN to predict century
2) We used the frequency of top k most common n-grams (consecutive n words) as attributes and used a simple ANN to predict century
3) We used word embeddings as attributes and a feedforward neural network to predict century
4) We used naïve bayes with bag of words approach
5) We used term frequency-inverse document frequency (TF-IDF) vectorizer to vectorize words(calculated by (# of times a word appears in a document / # of words in that document)*(total # of documents / # of documents containing this word)) and used linear SVM to predict century

Word sequence level:

We used word embeddings as attributes and used recurrent neural networks (RNN), long short term memory recurrent neural networks (LSTM), and bidirectional long short term memory neural networks (Bidirectional LSTM) to predict the century in which the poem was written.

Since simple ANN and naïve bayes have been examined extensively throughout the semester, we will not re-introduce the concepts here. RNN is a type of neural network in which each neuron's output is fed back as a part of the new input for the next time step. Although RNN's design makes it a power technique for sequence prediction, it has drawbacks such as attenuation of inputs from previous time steps and vanishing gradient problem. LSTM was therefore designed to cope with these problems by using the concept of long short term memory. In LSTM, apart from the usual memory state given by neuron's activation, it also has a cell state that functions as an explicit type of memory. LSTM also incorporates different gates at different levels to control the amount of information to filter out before passing the information to the next time step. While LSTM solved the major drawbacks of RNN, it can still only predict sequences based on information from the past. Bidirectional LSTM, as its name suggests allows the model to integrate information from both past and future for sequence prediction by having two copies of LSTM architecture that conducts forward propagation from both the beginning and the end of a sequence and combines the information from both direction for the model to learn. Bidirectional LSTM is the most complicated and arguably the most powerful recurrent neural network structure among the three we implemented. Support vector machine is an algorithm that aims to find a best line/plane/hyperplane to separate a group of data in vector space. When the dataset is not linearly separable, SVM uses kernels to transform the datapoints into higher dimensions to make linearly separable. In our implementation, we used SVM with a linear kernel, which doesn't transform dataset into higher dimensions and the performance will indicate how linearly separable the dataset is.

For this project, our goal was to cross compare the performance of all above implementations and implicate the features characteristic of the date of poem and how poem patterns evolve over centuries.

Experimental set up

We will now detail the experimental set up for all the implementations. The random seed used for all experiments was 30. All neural networks were implemented using keras with TensorFlow backend and had ReLU for hidden layer activation function, softmax for output layer activation function, cross entropy for loss function and trained using Adam optimizer. Word embeddings were implemented using keras and we tested word embedding dimensions using 64, 128, 256, and 512. For all the train, validate, test split, the training percentage was 0.7, validation percentage is 0.15, and test percentage is 0.15. We also examined the effect of stopwords removing and stemming on the performance.

Sentence structural level:

The ANN has one hidden layer with 20 hidden neurons. The number of input neurons is the number of attributes and the number of output neurons is the number of labels. The model was trained for 500 epochs, with a batch size of 128 with a learning rate of 0.01. The early stopping was set with a patience of 50. In terms of attributes measurement, the overall poem sentiment was measured using TextBlob from nltk library and the part of speech tagging was implemented using POS-tagger from nltk library.

Word frequency level:

Part 1 and 2 used the same ANN implementation as in sentence structural level. For part 3, the ANN was test using various number of hidden layers (1, 2, 3) and there was a dropout layer with 0.5 dropout rate between all layers. The model was trained for 500 epochs with a batch size of 128 with patience set to be 25 for early stopping. The learning rate was 0.01. For part4, we manually implemented naïve bayes. For part 5, word vectorization was implemented using TF-IDF vectorizer from scikit-learn. SVM was implemented using scikit-learn with linear kernel. In addition, word clouds for 17_{th},18_{th},19_{th} century poems was plotted using wordcloud library.

Word sequence level:

RNN, LSTM and bidirectional LSTM were all tested with different hidden layers (1,2,3), different embedding dimensions (64, 128, 256, 512), different hidden neuron numbers (128, 256, 512), and different learning rate (0.0001, 0.001, 0.01). Effect of stopwords removal and stemming was also explored with bidirectional LSTM. A dropout layer with 0.5 dropout rate were implemented between all layers of neural networks and the models were trained for 500 epochs with a batch size of 128 with patience set to be 25 for early stopping.

The performance of the algorithms was measured by accuracy and recall and precision.

Results

Due to the large amount of results generated using grid search, only a selective amount of results with best performances are listed below.

Sentence structure level:

ANN with sentence structural attributes

Table 2. Accuracy and CI for ANN with sentence structure attributes

Accuracy	CI
0.648	[0.605, 0.691]

Table 3. Confusion matrix for ANN with sentence structure attributes

	Predicted 17	Predicted 18	Predicted 19
Actual 17	181	30	13
Actual 18	44	116	45
Actual 19	32	50	97

Table 4. Recall and precision for ANN with sentence structure attributes

	17	18	19
Recalls	0.912	0.714	0.734
Precisions	0.853	0.738	0.768

From table 2, we can see that the accuracy is about 65% which is not very high. 17th century poem had both highest recall and precision. From the confusion matrix, we can see that the model had difficulty differentiating 18th vs 19th century poems. We also monitored the training process and the final accuracy on the training dataset was also below 70%, indicating that the sentence structure attributes we selected was not very predicative for the poem's date, especially not characteristic enough to differential between 18th and 19th century poems.

Word frequency level:

ANN with relative word frequencies

Table 5. Accuracy and CI for ANN with word frequencies as attributes

Traditional ANN	Accuracy	C.I.
using with top 200 words freq proportion	0.643	[0.600, 0.687]
using top 2000 words freq proportion	0.719	[0.678, 0.760]
using relative frequency of all words (remove stopwords)	**0.789**	**[0.752, 0.827]**
using relative frequency of all words (remove stopwords and did stemming)	0.74	[0.700, 0.780]
using relative frequency of all words (with stopwords)	0.73	[0.690, 0.771]

Table 6. Confusion matrix for ANN with all word frequencies

	Predicted 17	Predicted 18	Predicted 19
Actual 17	198	13	6
Actual 18	29	163	30
Actual 19	5	45	119

Table 7. Recall and precision for ANN with all word frequencies

	17	18	19
Recalls	0.808	0.567	0.542
Precisions	0.704	0.592	0.626

The results showed a positive correlation between the amount of words used and the performance accuracy. Notably, the model achieved highest accuracy (79%) when using all words' relative frequencies and the accuracy is marginally significantly higher than the accuracy obtained by using top 2000 words' relative frequencies. This indicated that word frequencies are a strong predictor of the century in which the poem was written. It was also interesting that stemming and stopwords removal didn't cause a statistically significant change of performance accuracy.

ANN with n-grams

Table 8. Accuracy and CI for ANN with relative frequencies of n-grams

Accuracy	Confidence Interval	n successive words	# most of frequent n-grams used
0.67	[0.627, 0.712]	2	10000
0.595	[0.551, 0.640]	3	10000
0.566	[0.521, 0.611]	3	5000
0.73	**[0.690, 0.771]**	**2**	**30000**
0.688	[0.645, 0.730]	2	5000
0.403	[0.358, 0.448]	4	2000
0.378	[0.334, 0.422]	5	1000

Table 9. ANN with top 30000 bigrams

	Predicted 17	Predicted 18	Predicted 19
Actual 17	198	15	11
Actual 18	28	106	71
Actual 19	14	27	138

Table 10. Recall and precision for ANN with top 30000 bigrams

	17	18	19
Recall	0.884	0.517	0.771
Precision	0.825	0.716	0.627

The results revealed an inverse correlation between the number of consecutive words n picked and the resulting accuracy. This is reasonable as the longer the n-grams get, the less likely it would occur and therefore longer n-grams are less characteristic. The best accuracy is achieved using the relative frequency of top 30000 bigrams, with an accuracy of 73%. This accuracy is marginally significantly higher than the accuracy obtained using top 10000 bigrams, again indicating the more word (pairs) we use, the more comprehensive and predictive it gets when dating poems.

ANN with word embedding approach

Table 11. Accuracy and CI for ANN with word embeddings

Accuracy	CI	Embedding	# Neurons	Learning Rate	Remove stopwords?	# hidden Layers	Drop out layers	Stemming?
0.69	[0.65, 0.73]	128	128	0.01	T	1	1	F
0.65	[0.61, 0.69]	128	128	0.01	T	2	2	F

Table 12. Confusion matrix for ANN with word embeddings with best performance

	Predicted 17	Predicted 18	Predicted 19
Actual 17	181	39	4
Actual 18	18	158	29
Actual 19	4	94	81

Table 13. Recall and precision for ANN with word embeddings with best performance

	17	18	19
Recall	0.808	0.771	0.453
Precision	0.892	0.543	0.711

The best performance achieved 69% accuracy, and again 78th century poems had highest recall and accuracy and the model tended to misclassify 19th century poems as written in 18th century. From what we tested, hyperparameters didn't affect the overall performance of this model much, which is understandable as this model is not very intricate.

Naïve Bayes

Table 14. Accuracy and CI for Naïve Bayes

Naïve Bayes	Stemming	No Stemming
No stopword removal	0.801, [0.769, 0.833]	0.804, [0.773, 0.836]
Stopword removal	0.775, [0.741, 0.808]	0.791, [0.759, 0.823]

Table 15. Confusion matrix for naïve bayes

	Predicted 17	Predicted 18	Predicted 19
Actual 17	194	25	5
Actual 18	10	159	36
Actual 19	2	43	134

Table 16. Recall and precision for naïve bayes

	17	18	19
Recalls	0.884	0.795	0.715
Precisions	0.93	0.706	0.78

The overall accuracy of naïve bayes was around 80%, which was very high. This gave a clear indication that the choice of words was very predictive of the century in which the poem was written. It is also worth noting that stemming and stopwords removal didn't affect the overall performance, gave further evidence for the distinctiveness of word choices across centuries.

Linear SVM

Table 17. Accuracy and CI for Linear SVM with TF-IDF word vectorization

SVM (linear kernel)	Stemming	No Stemming
No stopword removal	0.768, [0.745, 0.792]	0.782, [0.759, 0.806]
Stopword removal	0.777, [0.753, 0.800]	0.783, [0.760, 0.806]

Table 18. Confusion matrix for SVM

	Predicted 17	Predicted 18	Predicted 19
Actual 17	369	46	17
Actual 18	21	331	80
Actual 19	9	91	254

Table 19. Recall and precision for SVM

	17	18	19
Recall	0.854	0.766	0.718
Precision	0.925	0.707	0.724

Surprisingly, linear SVM achieved an overall high accuracy around 78%, indicating the vectorized dataset was highly linearly separable. Again, the stemming and stopwords removal didn't affect the performance significantly. Given that TF-IDF vectorizer calculates the relative frequency of a word in a poem weighted by the uniqueness of that word to that particular poem, the high performance of the model gives strong indication that word choices are characteristic of the century in which the poem was written.

Word sequence level:

RNN with word embedding approach

Table 20. Accuracy and CI for RNN with word embedding approach with different hyperparameters

Accuracy	Confidence Interval	Embedding	# of neurons	Learning rate	# of hidden layers	# dropout layers
0.363	[0.325, 0.402]	512	512	0.01	1	1
0.337	[0.301, 0.376]	512	512	0.01	2	2
0.365	[0.327, 0.403]	512	512	0.01	3	3
0.368	[0.330, 0.407]	128	128	0.01	1	1

Table 21. Confusion matrix for RNN

	Predicted 17	Predicted 18	Predicted 19
Actual 17	134	79	0
Actual 18	134	73	0
Actual 19	45	33	0

Table 22. Recall and precision for RNN

	17	18	19
Recall	0.629	0.353	0
Precision	0.428	0.395	0

Surprisingly, RNN's results were not statistically significantly higher than chance (33.3%), indicating that RNN model failed to learn the underlying relationship between word embeddings and the centuries in which the poem was written. It is worth noting that the model misclassified all 19th century poems into 17th and 18th centuries, and 17th century poems again had highest recall and precision.

LSTM with word embedding approach

Table 23. Accuracy and CI for LSTM with word embedding approach with different hyperparameters

Accuracy	Confidence Interval	Embedding	# of neurons	Learning rate	# of hidden layers	# of dropout layers
0.556	[0.516, 0.595]	512	512	0.01	1	1
0.64	[0.605, 0.681]	512	512	0.01	2	2
0.571	[0.53, 0.61]	512	512	0.01	3	3

Table 24. Confusion matrix for LSTM

	Predicted 17	Predicted 18	Predicted 19
Actual 17	169	43	12
Actual 18	20	134	51
Actual 19	8	83	88

Table 25. Recall and precision for LSTM

	17	18	19
Recall	0.754	0.654	0.492
Precision	0.858	0.515	0.583

The grid search results (which are not listed here due to the limit of space) yielded an average accuracy of 37%, and only working models are listed in the table with hyperparameters indicated. The best accuracy is 64% achieved by using 512 embedding dimension, 512 hidden neurons per layer, 2 hidden layers and 0.01 learning rate. The accuracy of the working models are statistically significantly higher than RNN models, agreeing with our expectation that LSTM would perform better than vanilla RNN models. However, the overall performance of

LSTM is still surprisingly low. Once again, the model is best at predicting 17th century poems.

Bidirectional LSTM with word embedding

Table 26. Accuracy and CI for bidirectional LSTM with word embedding with different hyperparameters

Accuracy	**CI**	Embedding	# Neurons	Learning Rate	Remove stopwords?	# Layers	Drop out layers	Stemming?
0.722	[0.686, 0.758]	128	128	0.01	T	1	1	F
0.66	[0.622, 0.697]	128	128	0.01	F	1	1	F
0.676	[0.639, 0.713]	128	128	0.01	T	1	1	T
0.7	[0.662, 0.735]	128	128	0.01	T	1	1	F

Table 27. Confusion matrix for bidirectional LSTM

	Predicted 17	Predicted 18	Predicted 19
Actual 17	201	20	3
Actual 18	30	150	25
Actual 19	6	81	92

Table 28. Recall and precision for bidirectional LSTM

	17	18	19
Recall	0.813	0.624	0.642
Precision	0.824	0.601	0.661

The best performance of bidirectional LSTM achieved a 72% accuracy, which is statistically significantly higher than the best performance of LSTM. From what we tested, bidirectional LSTM generally gives an accuracy that is close to 70%, which agrees with our expectation that bidirectional LSTM outperforms LSTM. The confusion matrix results still revealed that the model misclassified a lot of 19th century poems into 18th century poems.

Word Cloud Analysis

To visualize the top occurring unigrams characteristics of poems in each century, the word clouds are plotted in Figure 1.

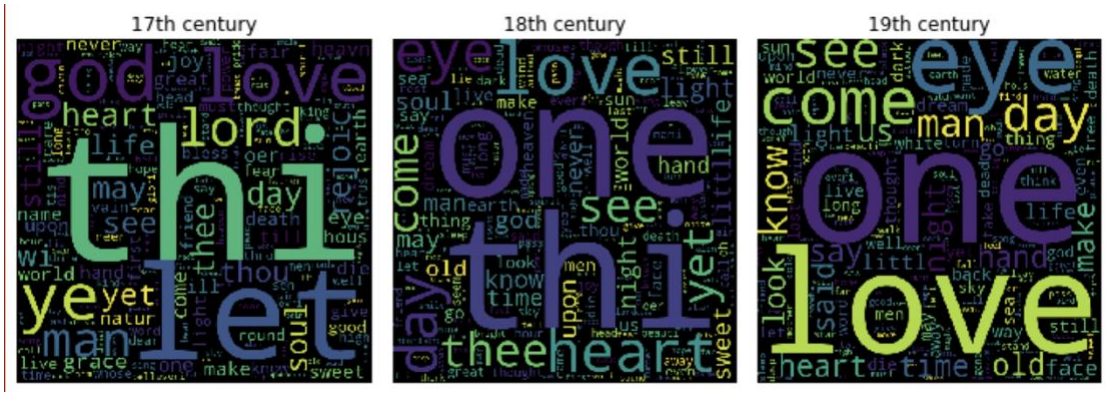

Figure 1. wordclouds for poems across centuries

From Figure 1, we can see that although love seem to be the common theme of all centuries, the focus seem to have shifted. Poems in 17th centuries have high frequency words such as "god", "lord", "thi", indicating the poems were written to pray the lord. In 18th century, the frequency of word "god" gets much lower and when it comes to 19th century, despite the fact that "love" became the highest occurring word, words such "god" or "lord" didn't show up at all. This potentially suggests a change of theme from "love of god" in a religious context to "the worldly love" in the popular culture.

Summary statistics and tables

Table 29. Accuracy and CI of the best performing model for each algorithm

ML Algorithm	Accuracy	Confidence Interval
RNN	0.416	[0.372, 0.459]
LSTM	0.640	[0.605, 0.681]
Sentence structure ANN	0.648	[0.605, 0.691]
Word embedding ANN	0.690	[0.650, 0.730]
Bidirectional iLSTM	0.722	[0.686, 0.758]
N-gram ANN	0.730	[0.690, 0.771]
Linear SVM	0.783	[0.760, 0.806]
Word frequency ANN	0.789	[0.752, 0.827]
Naive Bayes	0.804	[0.773, 0.836]

Table 30. Cross comparison of the performance of all algorithms

Machine Learning Algorithm	statistically significantly outperformed another approach	did statistically significantly worse than another approach	had statistically similar results
RNN	0	8	0
LSTM	1	5	2
Sentence Structure ANN	1	3	4
Word embedding ANN	1	3	4
Bidirectional LSTM	2	2	4
N-gram ANN	2	1	5
Linear SVM	5	0	3
Word Frequency ANN	4	0	4
Naive Bayes	6	0	2

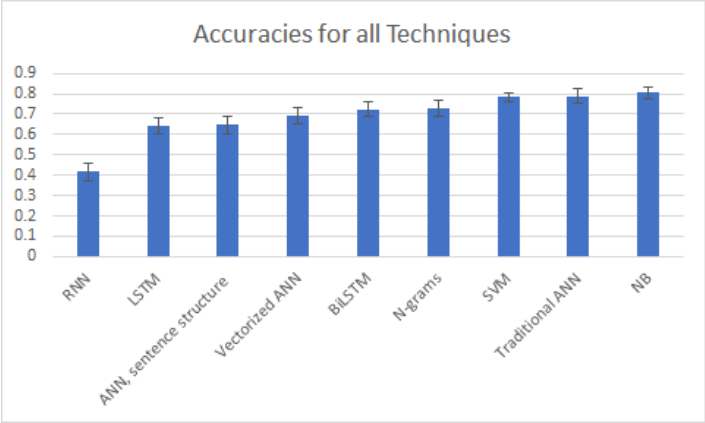

Figure 2. bar char for accuracies of all algorithms (CI indicated as error bars)

Results summary

Among all methods, the highest accuracy was achieved by using naïve bayes (~80%), linear SVM (~78%) and traditional ANN with relative frequencies of all words as attributes (~80%), indicating that choice of words are very characteristic of the century in which a poem was written. One outlier in the word frequency analysis level is the ANN with word embedding approach (65-70%). This is understandable because this approach doesn't tries to classify the vectorized words but unlike TF-IDF vectorizer, word embedding doesn't vectorize words solely based on their frequencies, but instead relies on other factors, such as the closeness of meanings of words. Therefore, ANN with word embedding approach is technically not a word frequency analysis approach, although it is associated with the choice of each individual word. The low performance of this approach indicates that the relative frequencies or the conditional probabilities of the usage of words is a cleaner measure of word choices and is more indicative of the date of the poem. Another approach that didn't achieve very high accuracy is the ANN with n-gram approach. The best performance of n-gram approach gave 73% accuracy, which indicates that the choice of unigram was of greater importance and word combinations were less indicative.

On the sentence structure level, ANN with structural attributes, such as poem length, overall sentiment, relative frequencies of part of speeches, etc. gave a 65% accuracy, which was not very high and indicates the sentence structures and overall sentiment were not very indicative of poems' dates.

The most intriguing part is the word sequence analysis approaches. It was expected that sequence prediction would be a reasonable approach for text classification and we anticipated RNN family of neural networks to achieve highest performance among all methods. However, the only working model in the RNN family is bidirectional LSTM and the overall accuracy is around 70% which was acceptable but not as great as the accuracies achieved by simpler models such naïve bayes or linear SVM. This indicated that word sequences were not very indicative of the century in which the poems were written.

Overall, these results indicated that word choices were more predictive of poems' date than sentence structures or word sequences. The complexity of bidirectional LSTM allows it to outperform other neural networks in the RNN family, but the results were still less optimal. The results also demonstrated that the concept of Occam's razor that sometimes simpler models would actually achieve better performance.

Conclusion and future work

We implemented multiple machine learning algorithms, including neural networks, naïve bayes and support vector machine to predict the centuries in which the poems were written. We implemented the algorithms that analyzed poems from sentence structural level, word frequencies/choices level, and word sequences level. We observed that highest frequencies were achieved by word choices analysis implementations using algorithms such as naïve bayes and linear SVM. Surprisingly, RNN models failed for this poem dating task and LSTM models' performances were suboptimal as well. Bidirectional LSTM did achieve acceptable performance but the accuracy was lower than some simpler methods such as naïve bayes and linear SVM. Overall, our results suggest that for our particular dataset, word choices were more indicative of poems' date than sentence structures or word sequences. Also, the results demonstrated that sometimes simpler models may outperform more complicated ones, especially for simple tasks.

In terms of our future work, we would like to examine whether lexical stress [1] could be a good predictor for poems' date. We would like to try different kernels for SVM, such as polynomial kernel, or radial kernel to see if transforming the data into higher spaces would further improve the performance by making the data more separable. One limitation about our dataset is that the dataset only included the birth and death of the authors and the exact date of poetry were missing. We tackled this by using the midpoint of the authors' life spans as the date of the poetry and this could be quite inaccurate and may explain why most models struggled to differential between 18th and 19th century poems. Our next step would be find more comprehensive and accurate dataset and rerun the experimentation, especially for RNN family of neural networks to see if the failure of these neural networks were caused by the inaccurate labeling of poems.

References

1. Agirrezabal, M., Alegria, I., & Hulden, M. (2016). Machine Learning for Metrical Analysis of English Poetry. *Machine Learning for Metrical Analysis of English Poetry*, *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 772–781. Retrieved from https://www.aclweb.org/anthology/C16-1074.pdf

2. Alsharif, O., Alshamaa, D., & Ghneim, N. (2013). Emotion Classification in Arabic Poetry using Machine Learning. *International Journal of Computer Applications International Journal of Computer Applications*, *65*(March), 10–15. Retrieved from https://pdfs.semanticscholar.org/9d24/0ab714a97b6d06c094bdba1a41e7ec6b33b5.pdf

3. Gluck, C. (2017, September 28). When was this poem written? My computer can tell you. Retrieved October 30, 2019, from https://towardsdatascience.com/identifying-when-poems-were-written-with-natural-language-processing-a40ff286bcd.

4. Gluck, C. (2019, March 14). chaimgluck/projects. Retrieved November 15, 2019, from https://github.com/chaimgluck/projects/tree/master/Project-poems?fbclid=IwAR0AnlI1BnoJbB8b6iCr9jp4vJKYGP8v_H7lEpOXcqGgnNAPRz184Ig72vg.

5. Cambria, E., & White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. *IEEE Computational Intelligence Magazine*, *9*(2), 48–57. doi: 10.1109/mci.2014.2307227

We affirmed that we have adhered to the Honor Code in this assignment.

---- Jiachen Liu, Simon Harris, Will Tokunaga